Microsoft Windows

Protected Processes

November 27, 2006

Abstract

The Microsoft® Windows Vista™ operating system introduces a new type of process known as a *protected process* to enhance support for Digital Rights Management functionality in Windows Vista.

This paper describes the characteristics of protected processes, and how other software on the system can and cannot interact with these processes. This document is for developers of products that monitor and report on other processes in the system, so that they can understand the unique constraints involved when interacting with a protected process.

The current version of this paper is maintained on the Web at:
http://www.microsoft.com/whdc/system/vista/process_Vista.mspx

Contents

Disclaimer

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Introduction

The Microsoft® Windows Vista™ operating system introduces a new type of process known as a *protected process* to enhance support for Digital Rights Management functionality in Windows Vista. These protected processes exist alongside other processes in Windows Vista.

The primary difference between a typical process and a protected process is the level of access that other processes in the system can obtain to protected processes. In versions of Microsoft Windows® earlier than Windows Vista, the process model allows a parent process to acquire a handle to and manipulate the state of any child process that it creates. Likewise, processes that users created with sufficient privileges can access and manipulate the state of all processes on the system. This behavior remains true for typical processes. However, the level of access to protected processes and threads within those processes is significantly more constrained.

Any application can attempt to create a protected process. However, due to the restrictions of running inside a protected process, the operating system requires that these processes be specially signed. In Windows Vista, Protected Media Path (PMP) uses the protected process infrastructure to provide increased protection for high-value media content. Developers can leverage protected processes by using the Media Foundation API. For more information, see Output Content Protection and Windows Vista.

This paper describes the characteristics of protected processes and how other software on the system can and cannot interact with these processes. This paper is for developers of products that monitor and report on other processes in the system, so that they can understand the unique constraints involved when interacting with a protected process.

Examples of Constraints with Protected Processes

Developers who are used to interacting with typical processes will notice the following significant differences, among others, in interacting with protected processes.

Constraints on protected processes. A typical process cannot perform operations such as the following on a protected process:

- Inject a thread into a protected process
- Access the virtual memory of a protected process
- Debug an active protected process
- Duplicate a handle from a protected process
- Change the quota or working set of a protected process

Note

Certain process properties (such as the working set) can be indirectly changed through membership in a job object.

Constraints on the threads of a protected process. A typical process cannot perform the following operations such as the following on the threads of a protected process:

- Set or retrieve context information
- Impersonate the thread

Process and Thread Access Rights

An access mask determines the specific access rights that are enabled for each process or thread. The following tables show the entire set of access rights for processes and threads and specifies the subset of access rights that are allowed from a typical process to a protected process and the threads within a protected process.

Certain access rights provided problems for the design of protected processes. Windows Vista introduces *limited* versions of these access rights that provide a access to a subset of the information that the original versions accessed.

Table 1. Process Access Rights

Access right	Meaning	Allowed
Standard		
DELETE	Required to delete the object.	No
READ_CONTROL	Required to read information in the security descriptor for the object, not including the information in the system access control list (SACL).	No
SYNCHRONIZE	Required to use the process handle in any of the wait functions.	Yes
WRITE_DAC	Required to modify the discretionary access control list (DACL) in the security descriptor for the object.	No
WRITE_OWNER	Required to change the owner in the security descriptor for the object.	No
Process-specific		
PROCESS_ALL_ACCESS	All possible access rights for a process object. Required to debug an active process.	No
PROCESS_CREATE_PROCESS	Required to create a process.	No
PROCESS_CREATE_THREAD	Required to create a thread.	No
PROCESS_DUP_HANDLE	Required to duplicate a handle.	No
PROCESS_QUERY_INFORMATION	Required to read certain information from the process object.	No
PROCESS_QUERY_LIMITED_INFORMATION	Required to read certain information from the process object. This is a subset of the information provided by PROCESS_QUERY_INFORMATION.	Yes
PROCESS_SET_QUOTA	Required to set memory limits.	No
PROCESS_SET_INFORMATION	Required to set certain information in the process object.	No
PROCESS_TERMINATE	Required to terminate a process.	Yes
PROCESS_VM_OPERATION	Required to perform a memory	No

Access right	Meaning	Allowed
	operation in the address space of a process.	
PROCESS_VM_READ	Required to read memory in a process by using **ReadProcessMemory**.	No
PROCESS_VM_WRITE	Required to write to memory in a process by using **WriteProcessMemory**.	No

Table 2. Thread Access Rights

Access right	Meaning	Allowed
THREAD_ALL_ACCESS	All possible access rights for a thread object.	No
THREAD_DIRECT_IMPERSONATION	Required for a server thread to impersonate a client.	No
THREAD_GET_CONTEXT	Required to read the context of a thread.	No
THREAD_IMPERSONATE	Required to use a thread's security information directly by using a communication mechanism that provides impersonation services.	No
THREAD_QUERY_INFORMATION	Required to read certain information from the thread object.	No
THREAD_QUERY_LIMITED_INFORMATION	Required to read certain information from the thread object. This is a subset of the information provided by THREAD_QUERY_INFORMATION.	Yes
THREAD_SET_CONTEXT	Required to write to the context of a thread.	No
THREAD_SET_INFORMATION	Required to set certain information in the thread object.	No
THREAD_SET_LIMITED_INFORMATION	Required to set certain information in the thread object. This is a subset of the information provided by THREAD_SET_INFORMATION.	Yes
THREAD_SET_THREAD_TOKEN	Required to set the impersonation token for a thread.	No
THREAD_SUSPEND_RESUME	Required to suspend or resume a thread.	Yes
THREAD_TERMINATE	Required to terminate a thread.	No

Best Practices

Historically, a privileged service (running as administrator or local system) has been able to obtain all access to a process or thread, regardless of its DACL, by using **SeDebugPrivilege**.

Starting with Windows Vista, the privileges that are marked with “No” in the tables earlier in this paper cannot be obtained for a protected process or thread. This can be a problem if memory scanning is critical to the operation of the application.

Do not attempt to circumvent this restriction by installing a kernel-mode component to access the memory of a protected process because the system and third-party applications may rely on the fact that protected processes are signed code that is run in a contained environment.

Resources

Process Security and Access Rights
http://msdn.microsoft.com/library/en-us/dllproc/base/process_security_and_access_rights.asp